

EXHIBIT D

U.S. Patent No. 7,774,740 – Capital One Financial Corp., Capital One, National Association

Claim 11

Implicit, LLC (“Implicit”) provides evidence of infringement of claim 11 of U.S. Patent No. 7,774,740 (hereinafter “the ’740 patent”) by Capital One Financial Corp. and Capital One, National Association (collectively, “Capital One”). In support thereof, Implicit provides the following claim charts.

“Accused Instrumentalities” as used herein refers to at least the components involved in the dynamic generation of applications/applets/apps, including but not limited to webpages and mobile apps (or components thereof), which are dynamically generated by one or more Capital One servers running Node.js and V8 software in response to a request from a client device (e.g., a browser or mobile device used by a Capital One customer/subscriber/user to access data/information stored by Capital One) to the one or more Capital One servers. An example of the Accused Instrumentalities are the one or more servers that generate a response to a web-browser request directed to a Capital One webpage (e.g., <https://www.capitalone.com/>). These claim charts demonstrate Capital One’s infringement by comparing each element of the asserted claims to corresponding components, aspects, and/or features of the Accused Instrumentalities. These claim charts are not intended to constitute an expert report on infringement. These claim charts include information provided by way of example, and not by way of limitation.

The analysis set forth below is based only upon information from publicly available resources regarding the Accused Instrumentalities, as Capital One has not yet provided any non-public information. An analysis of Capital One’s (or other third parties’) technical documentation and/or software source code may assist in fully identifying all infringing features and functionality. Accordingly, Implicit reserves the right to supplement this infringement analysis once such information is made available to Implicit. Furthermore, Implicit reserves the right to revise this infringement analysis, as appropriate, upon issuance of a court order construing any terms recited in the asserted claims.

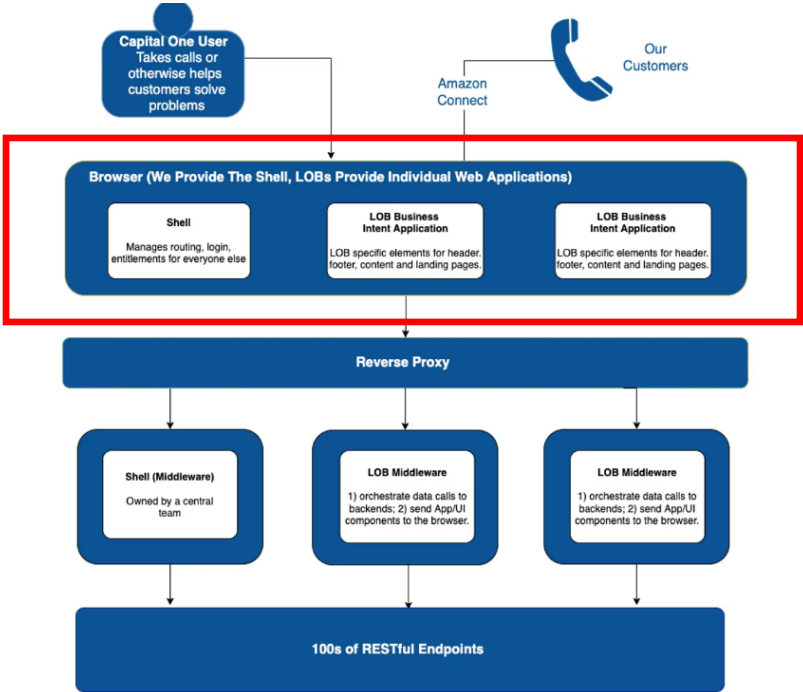
Implicit provides this evidence of infringement and related analysis without the benefit of claim construction or expert reports or discovery. Implicit reserves the right to supplement, amend or otherwise modify this analysis and/or evidence based on any such claim construction or expert reports or discovery.

Unless otherwise noted, Implicit contends that Capital One directly infringes the ’740 patent in violation of 35 U.S.C. § 271(a) by selling, offering to sell, making, using, and/or importing the Accused Instrumentalities. The following exemplary analysis demonstrates that infringement.

Unless otherwise noted, Implicit believes and contends that each element of each claim asserted herein is literally met through Capital One’s provision of the Accused Instrumentalities. However, to the extent that Capital One attempts to allege that any asserted claim element is not literally met, Implicit believes and contends that such elements are met under the doctrine of equivalents. More specifically, in its investigation and analysis of the Accused Instrumentalities, Implicit did not identify any substantial differences between the elements of the patent claims and the corresponding features of the Accused Instrumentalities, as set forth herein. In each instance, the identified feature of the Accused Instrumentalities performs at least substantially the same function in substantially the same way to achieve substantially the same result as the corresponding claim element.

IMPLICIT LLC'S ANALYSIS OF INFRINGEMENT

To the extent the chart of an asserted claim relies on evidence about certain specifically-identified Accused Instrumentalities, Implicit asserts that, on information and belief, any similarly-functioning instrumentalities also infringe the charted claim. Implicit reserves the right to amend this infringement analysis based on other products made, used, sold, imported, or offered for sale by Capital One. Implicit also reserves the right to amend this infringement analysis by citing other claims of the '740 patent, not listed in the claim chart, that are infringed by the Accused Instrumentalities. Implicit further reserves the right to amend this infringement analysis by adding, subtracting, or otherwise modifying content in the "Accused Instrumentalities" column of each chart.

Claim #	Accused Instrumentalities
<p>11. A method operating on a computer system, having a client computer and a server computer, for managing requests to the server computer, the method comprising:</p>	<p>The Accused Instrumentalities perform a method operating on a computer system, having a client computer and a server computer, for managing requests to the server computer. This is illustrated, for example, as follows:</p>  <p>The diagram illustrates the front-end architecture. At the top, 'Capital One User' (who takes calls or helps customers solve problems) and 'Our Customers' (via 'Amazon Connect') are shown. Arrows from both point to a red-bordered box labeled 'client'. This box contains a 'Browser (We Provide The Shell, LOBs Provide Individual Web Applications)' which includes a 'Shell' (managing routing, login, entitlements) and two 'LOB Business Intent Applications' (providing specific elements for header, footer, content, and landing pages). Below the browser is a 'Reverse Proxy'. Arrows from the browser and proxy point to three 'LOB Middleware' boxes (each orchestrating data calls to backends and sending App/Ui components to the browser). These point to a final box labeled '100s of RESTful Endpoints'. The entire flow from the client to the endpoints is labeled 'server (backend)' in red text.</p> <p>Our front-end architecture matches pretty well to what Michael Geers might characterize as an “app shell” with multi-level routing. At the foundational level, we lean heavily on standard open source software libraries and practices. For backend libraries, we use (among others) Fastify, NestJS, Pino, Restify and Undici. On the front-end, we utilize Vue.js and React.js.</p> <p>All of our browser to backend connections go through Node microservices. These microservices are responsible for most of the middle-tier work you would expect in a distributed application. This includes data orchestration, business rule enforcement, and logical decisioning.</p> <p>Source: https://www.capitalone.com/tech/software-engineering/loosely-coupled-micro-frontends-with-nodejs/</p>

at the server computer, receiving a request from the client computer, the request identifying an application and identifying a form of the application; and	<p>The Accused Instrumentalities perform a method that includes, at the server computer, receiving a request from the client computer, the request identifying an application and identifying a form of the application. This is illustrated, for example, as follows:</p> <div><div>▼ Request Headers</div><table><tr><td>:authority:</td><td>www.capitalone.com</td><td>application</td></tr><tr><td>:method:</td><td>GET</td><td></td></tr><tr><td>:path:</td><td>/credit-cards/fair-and-building/</td><td></td></tr><tr><td>:scheme:</td><td>https</td><td></td></tr><tr><td>Accept:</td><td>text/html,application/xhtml+xml,application/x-exchange;v=b3;q=0.7</td><td></td></tr><tr><td>Accept-Encoding:</td><td>gzip, deflate, br</td><td>form</td></tr><tr><td>Accept-Language:</td><td>en-US,en;q=0.9</td><td></td></tr></table></div> <p>Source: https://www.capitalone.com/credit-cards/fair-and-building/</p>	:authority:	www.capitalone.com	application	:method:	GET		:path:	/credit-cards/fair-and-building/		:scheme:	https		Accept:	text/html,application/xhtml+xml,application/x-exchange;v=b3;q=0.7		Accept-Encoding:	gzip, deflate, br	form	Accept-Language:	en-US,en;q=0.9	
:authority:	www.capitalone.com	application																				
:method:	GET																					
:path:	/credit-cards/fair-and-building/																					
:scheme:	https																					
Accept:	text/html,application/xhtml+xml,application/x-exchange;v=b3;q=0.7																					
Accept-Encoding:	gzip, deflate, br	form																				
Accept-Language:	en-US,en;q=0.9																					
in response to receiving the request:	<p>The Accused Instrumentalities perform the method steps recited hereinafter in response to receiving the aforementioned request. This is illustrated below, for example, as follows:</p>																					

	<div data-bbox="478 196 1486 451"> <div>▼ General</div> <div>Request URL: https://www.capitalone.com/credit-cards/fair-and-building/</div> <div>Request Method: GET</div> <div>Status Code: ● 200</div> <div>Remote Address: 23.212.250.71:443</div> <div>Referrer Policy: origin-when-cross-origin</div> </div> <p>Source: https://www.capitalone.com/credit-cards/fair-and-building/</p>
<p>compiling the application into a compiled form; and</p>	<p>The Accused Instrumentalities perform the step of compiling the application into a compiled form. This is illustrated, for example, as follows with implementation of the Node.js v8 module:</p> <div data-bbox="527 732 1831 1036"> <div>Node.js v17.9.0 ► Table of contents ► Index ► Other versions ► Options</div> <div>Source Code: lib/v8.js</div> <p>The v8 module exposes APIs that are specific to the version of V8 built into the Node.js binary. It can be accessed using:</p> <pre>const v8 = require('v8');</pre> </div> <p>Source: https://nodejs.org/api/all.html#all_v8_v8</p>

	<div data-bbox="594 201 1761 625" data-label="Complex-Block"> <h3>Compilation</h3> <p>JavaScript is generally considered an interpreted language, but modern JavaScript engines no longer just interpret JavaScript, they compile it.</p> <p>This has been happening since 2009, when the SpiderMonkey JavaScript compiler was added to Firefox 3.5, and everyone followed this idea.</p> <p>JavaScript is internally compiled by V8 with just-in-time (JIT) compilation to speed up the execution.</p> </div> <p>Source: https://nodejs.dev/learn/the-v8-javascript-engine</p> <p>Upon information and belief, Capital One servers utilize Node.js and V8 engine (<i>see</i> https://developer.salesforce.com/blogs/2021/01/what-is-node-js-and-why-does-it-matter-as-a-salesforce-developer). Upon information and belief Node.js uses the V8 engine to compile code into the compiled form (<i>see</i> https://nodejs.dev/learn/the-v8-javascript-engine).</p>
transforming the compiled application into a transformed form of the compiled form of the application, wherein transforming comprises execution and compression of the compiled form;	<p>The Accused Instrumentalities perform the step of transforming the compiled application into a transformed form of the compiled form of the application, wherein transforming comprises execution and compression of the compiled form. This is illustrated, for example, as follows:</p>

Node.js v6.3.0 Documentation

[Index](#) | [View on single page](#) | [View as JSON](#)

Table of Contents

- **Zlib**
 - [Compressing HTTP requests and responses](#)
 - [Memory Usage Tuning](#)

Source: <https://nodejs.org/dist/v6.3.0/docs/api/zlib.html>

Zlib

Transforming

Stability: 2 - Stable

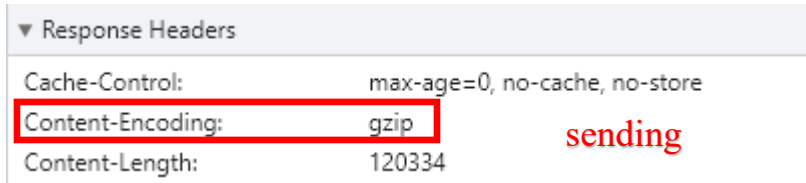
The `zlib` module provides compression functionality implemented using Gzip and Deflate/Inflate. It can be accessed using:

```
const zlib = require('zlib');
```

Compressing or decompressing a stream (such as a file) can be accomplished by piping the source stream data through a `zlib` stream:

```
const gzip = zlib.createGzip();
const fs = require('fs');
const inp = fs.createReadStream('input.txt');
const out = fs.createWriteStream('input.txt.gz');
```

Source: <https://nodejs.org/dist/v6.3.0/docs/api/zlib.html>

	Upon information and belief, Node.js uses Zlib for transforming the compiled form comprising execution and compression of the compiled form (<i>see</i> https://nodejs.org/dist/v6.3.0/docs/api/zlib.html .)
sending the transformed form of the application to the client computer.	<p>The Accused Instrumentalities perform the step of sending the transformed form of the application to the client computer. This is illustrated, for example, as follows:</p>  <p>The screenshot shows a 'Response Headers' section with three entries: 'Cache-Control: max-age=0, no-cache, no-store', 'Content-Encoding: gzip' (highlighted with a red box), and 'Content-Length: 120334'. The word 'sending' is written in red to the right of the 'Content-Encoding' header.</p> <p>Source: https://www.capitalone.com/credit-cards/fair-and-building/</p> <p>Upon information and belief, the transformed form 'gzip' is sent to the client computer (<i>see</i> https://www.capitalone.com/credit-cards/fair-and-building/).</p>

Caveat: The notes and/or cited excerpts utilized herein are set forth for illustrative purposes only and are not meant to be limiting in any manner. For example, the notes and/or cited excerpts, may or may not be supplemented or substituted with different excerpt(s) of the relevant reference(s), as appropriate. Further, to the extent any error(s) and/or omission(s) exist herein, all rights are reserved to correct the same.